객체지향 개발 방법론

DVM Project OOI

#4조

201710240 이해림

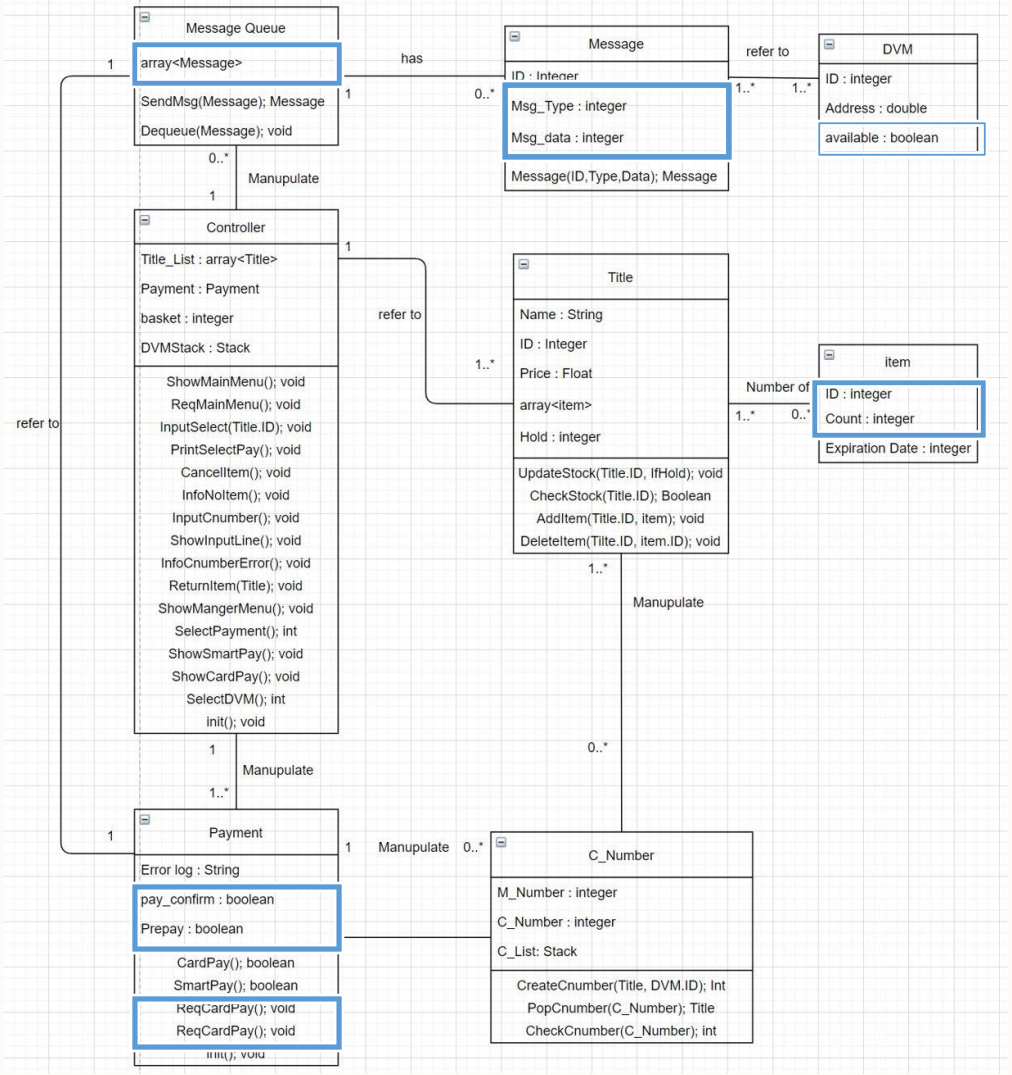
201811217 이해인

201711836 송호영

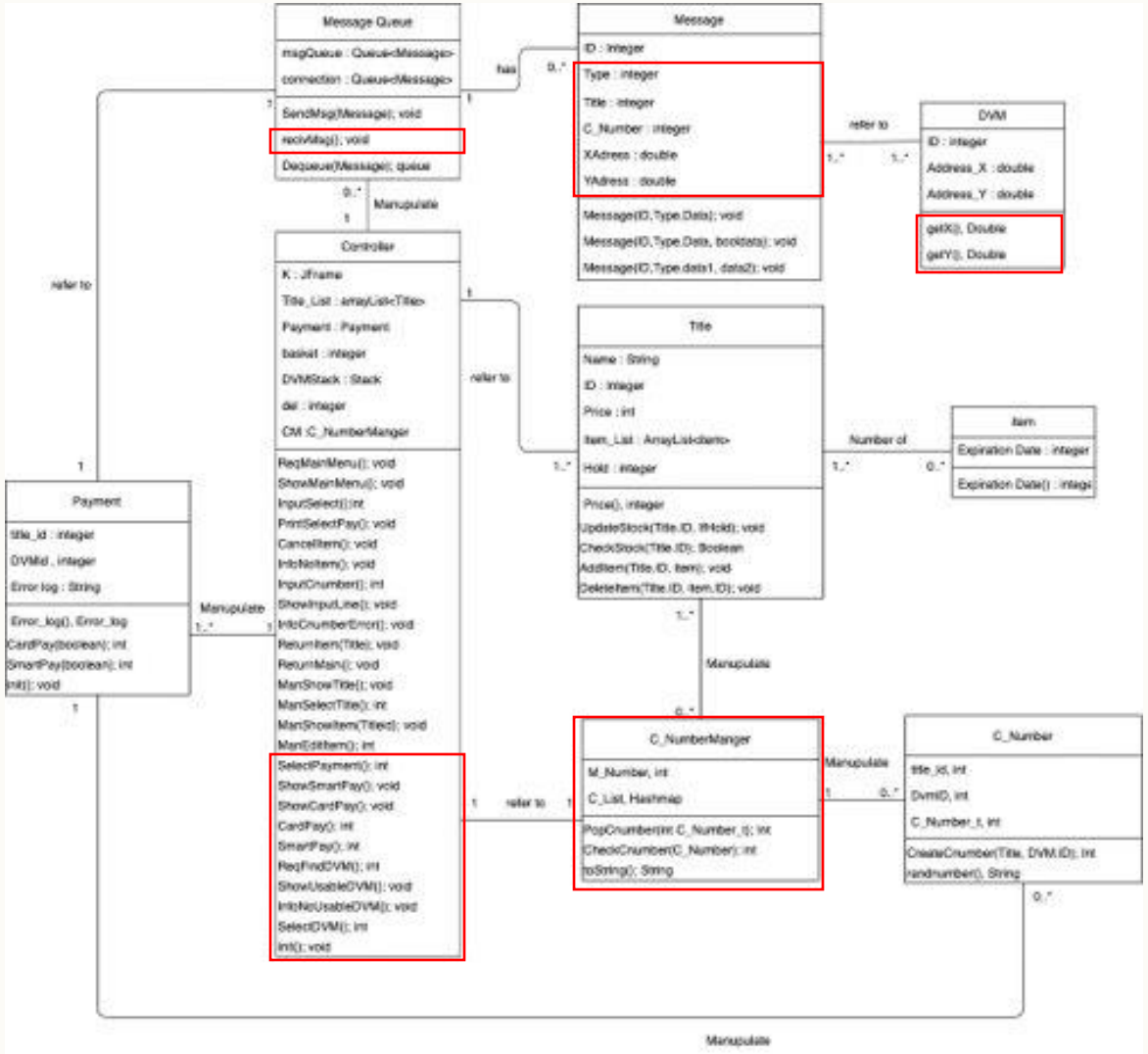
Index

Activity 2051. Implement Class & Methods Definitions

수정 전



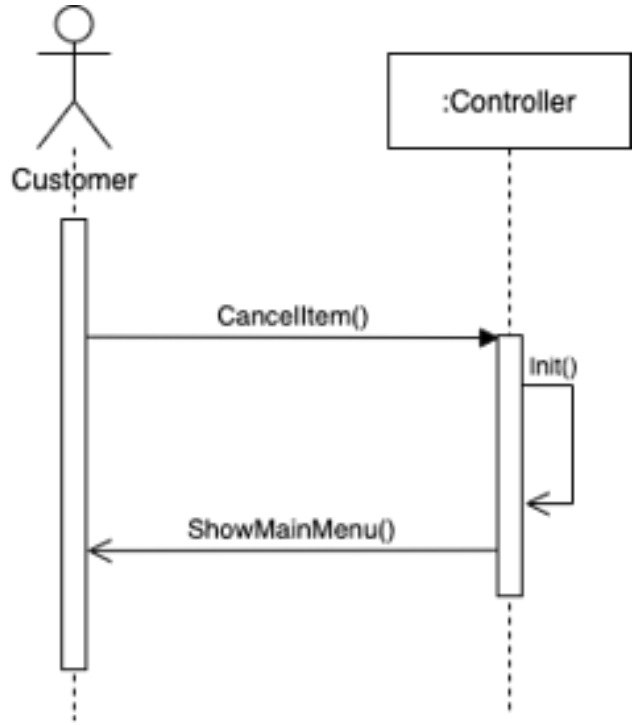
수정 후



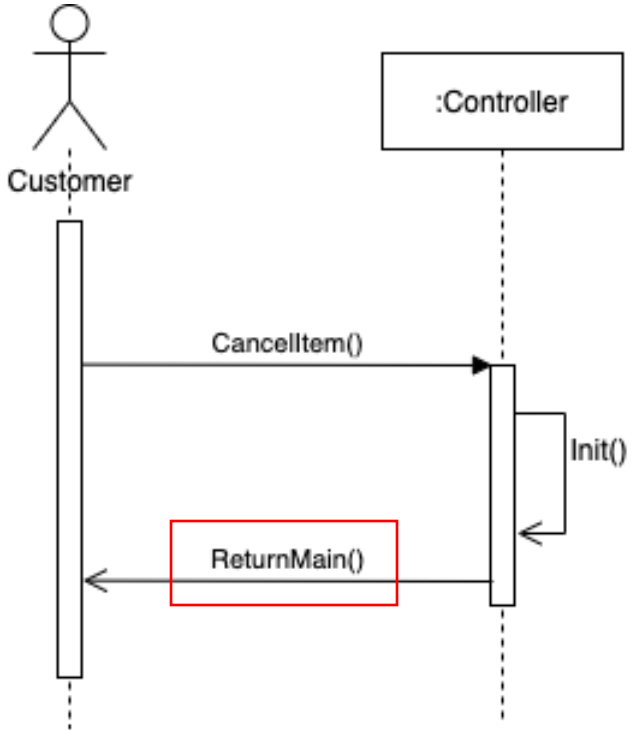
Activity 2051. Implement Class & Methods Definitions

Use Case 3. 사용자 선택 취소

수정 전



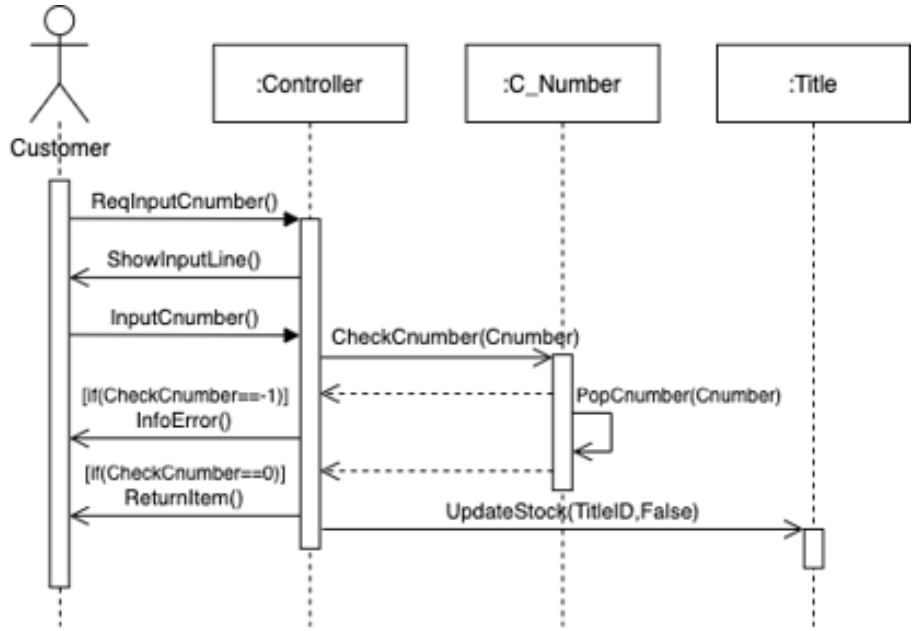
수정 후



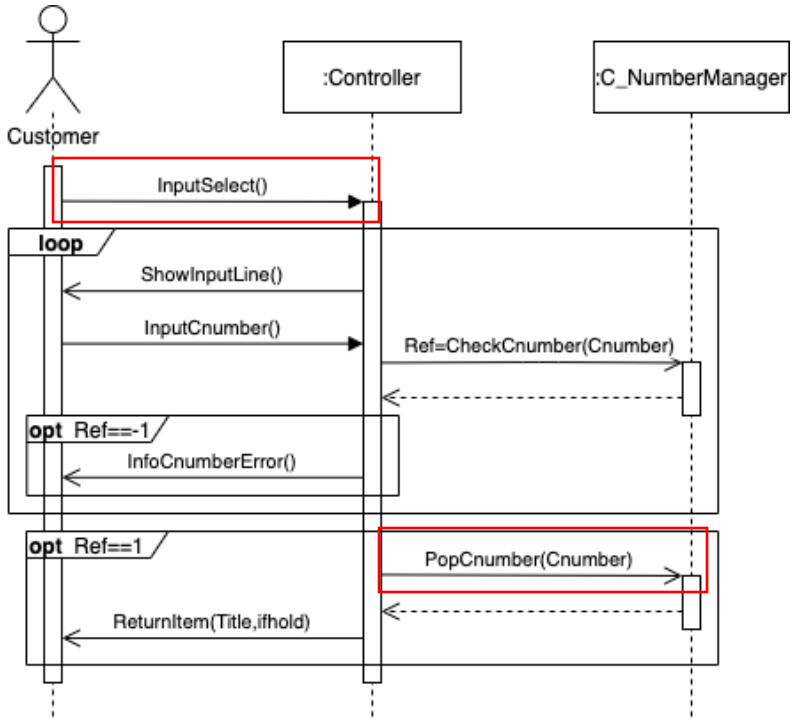
Activity 2051. Implement Class & Methods Definitions

Use Case

수정 전



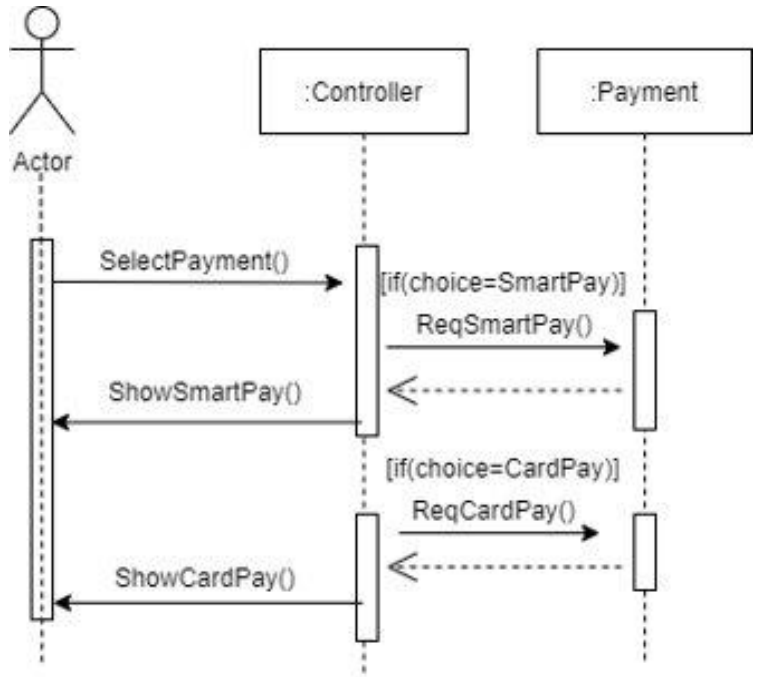
수정 후



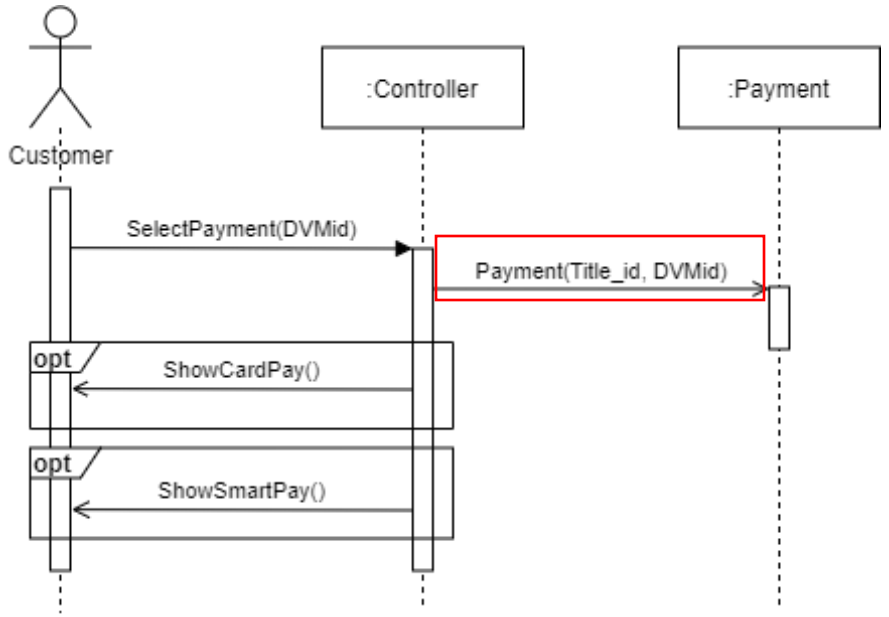
Activity 2051. Implement Class & Methods Definitions

Use Case

수정 전



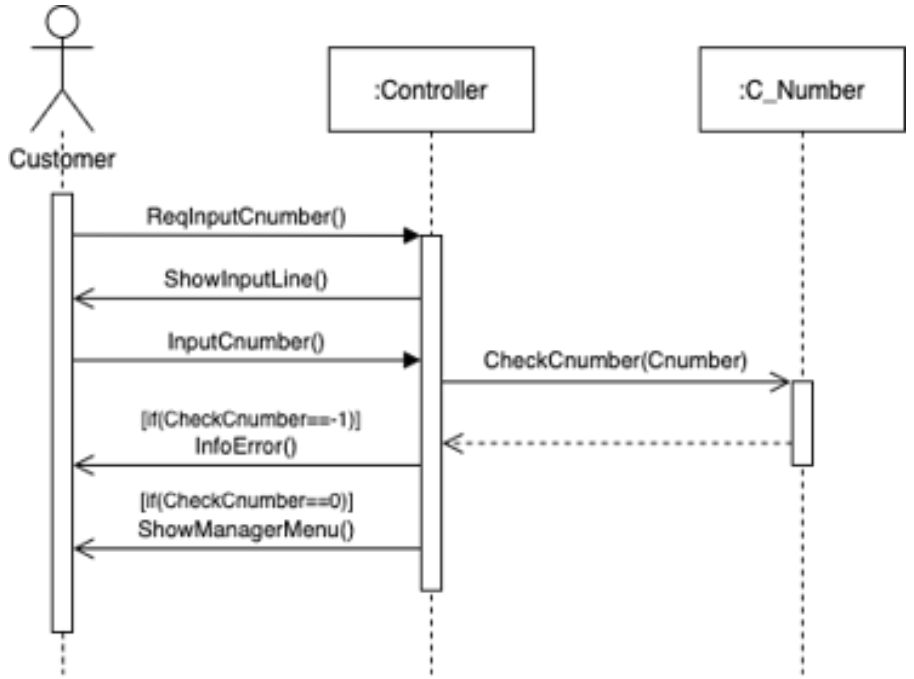
수정 후



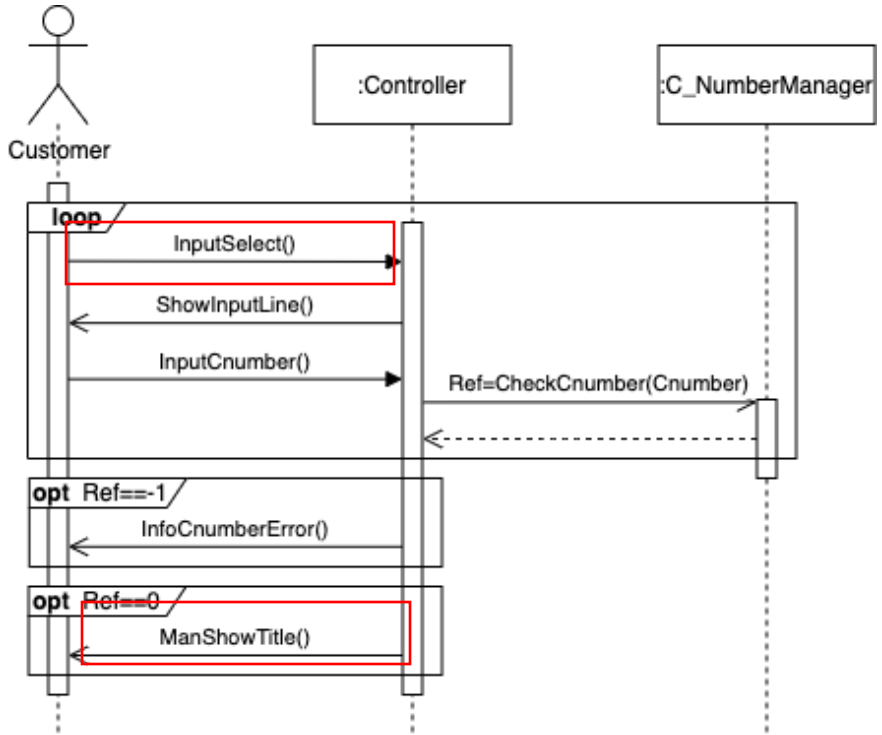
Activity 2051. Implement Class & Methods Definitions

Use Case

수정 전



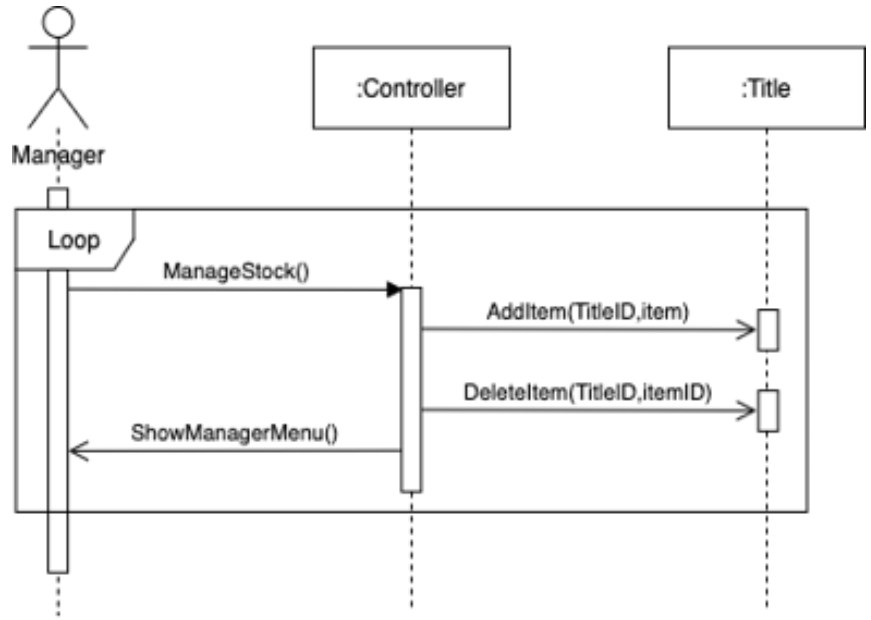
수정 후



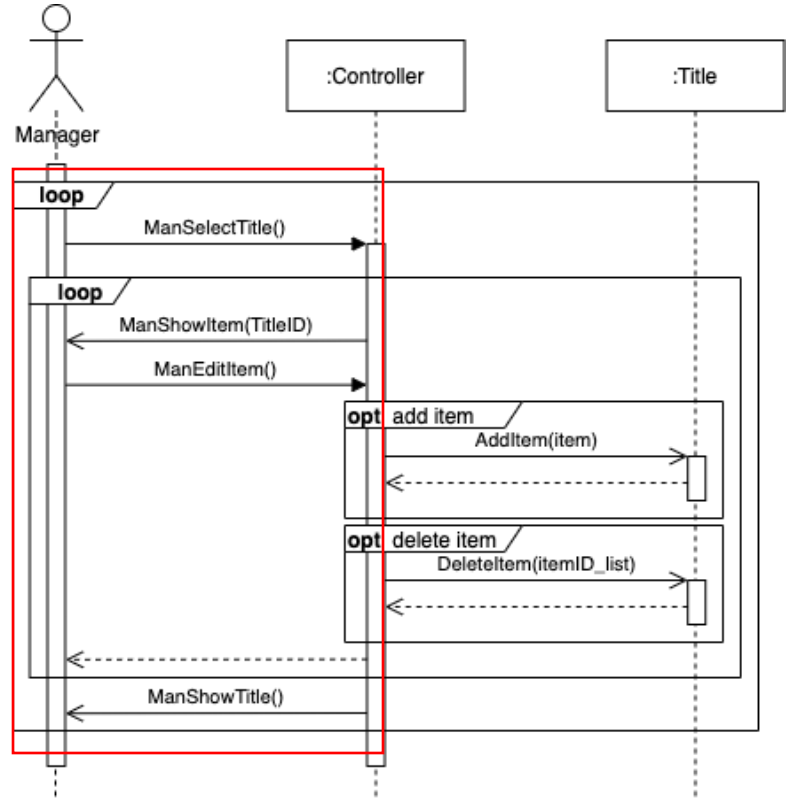
Activity 2051. Implement Class & Methods Definitions

Use Case

수정 전



수정 후



Activity 2051. Implement Class & Methods Definitions

Name	Controller
Type	Class
Purpose	사용자에게 메뉴를 표시
Overview	N/A
Cross Reference	R1.1, R1.2.1, R1.2.2, R1.2.3, R1.3, R1.4, R1.5, R2.1, R2.4, R3.1.2 "자판기 음료 종류 출력", "구매할 음료 입력", "사용자 선택 취소", "사용자 인증번호/바코드 입력", "재고가 부족한 재품 안내", "구매가능한 자판기 안내", "사용자 자판기 선택", "결제 수단 목록 출력", "인증번호/바코드 출력"
Input	TitleID, DVMid
Output	Integer , void
Exceptional Courses of Events	N/A

Activity 2051. Implement Class & Methods Definitions

Name	Title
Type	Class
Purpose	재고를 관리하고 그에 대한 정보를 저장하는 클래스
Overview	N/A
Cross Reference	R1.1, R1.2.1, R1.3, R1.4, R2.3.1, R2.3.2, R3.1.1, R3.2, R4.2, R4.4 "자판기 음료 종류 출력", "구매할 음료 입력", "재고가 부족한 재품 안내", "구매가능한 자판기 안내", "음료 전달", "재고 수량 업데이트", "인증번호/바코드 생성", "재고변경 및관리"
Input	Id, IfHold, item, ilst
Output	Name, Item_List, boolean
Exceptional Courses of Events	N/A

Activity 2051. Implement Class & Methods Definitions

Name	Item
Type	Class
Purpose	재고 각각의 정보를 저장하는 클래스
Overview	N/A
Cross Reference	R1.1, R1.2.1, R1.3, R1.4, R3.1.1, R3.2, R4.2, R4.4 "자판기 음료 종류 출력", "구매할 음료 입력", "재고가 부족한 재품 안내", "구매가능한 자판기 안내", "음료 전달", "재고 수량 업데이트", "인증번호/바코드 생성", "재고변경 및관리"
Input	N/A
Output	Expiration_date
Exceptional Courses of Events	N/A

Activity 2051. Implement Class & Methods Definitions

Name	Payment
Type	Class
Purpose	진행되는 결제정보를 컨트롤러에 전달하는 클래스
Overview	N/A
Cross Reference	R2.1, R2.3.1, R2.3.2, R2.4, R3.1.1 "결제 수단 목록 출력", "간편 결제", "카드 결제", "결제 취소", "인증번호, 바코드 출력"
Input	Title_id, DVMid, Pay_confirm
Output	Error_log, CN, integer
Exceptional Courses of Events	N/A

Activity 2051. Implement Class & Methods Definitions

Name	**C_NumberManager**
Type	Class
Purpose	인증번호 구매 진행시 저장된 인증번호를 확인하는 클래스
Overview	N/A
Cross Reference	R1.2.3, R3.1.1, R3.1.2 "사용자 인증번호/바코드 입력", "인증번호/바코드 생성", 인증번호/바코드 출력"
Input	Title_id, id, C_Number_t
Output	R, integer, String
Exceptional Courses of Events	N/A

Activity 2051. Implement Class & Methods Definitions

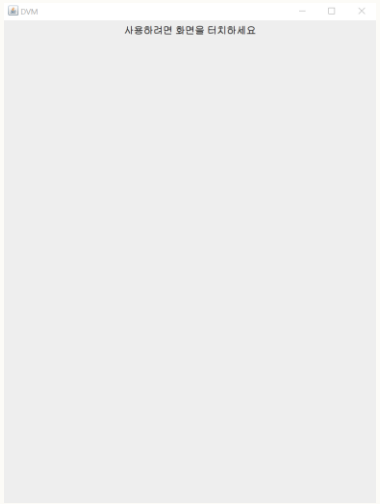
Name	C_Number
Type	Class
Purpose	인증번호를 생성하는 클래스
Overview	N/A
Cross Reference	R3.1.1, R3.1.2 "인증번호/바코드 생성", 인증번호/바코드 출력"
Input	Title_id, DvmID
Output	C_Number_t, numStr
Exceptional Courses of Events	N/A

Activity 2051. Implement Class & Methods Definitions

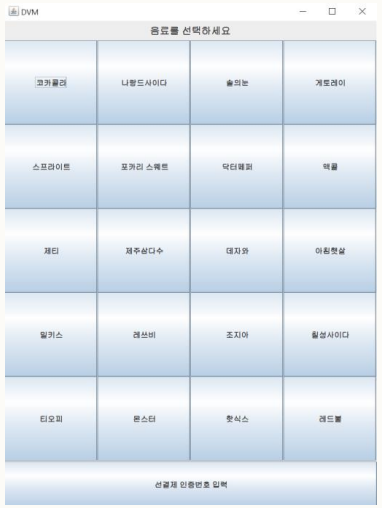
Name	**Message_Queue**
Type	Class
Purpose	메세지를 수신/발신하고 큐에 저장하는 클래스
Overview	N/A
Cross Reference	R1.4, R3.1.1, R4.3, R4.4, R4.5.1, R4.5.2 "구매가능한 자판기 안내", "인증번호/바코드 생성", "위치 정보 확인", "재고 정보 확인", "메시지 발신", "메시지 수신"
Input	Message, type
Output	temp1
Exceptional Courses of Events	N/A

Activity 2052. Implement Windows (사용자 매뉴얼)

첫 실행화면

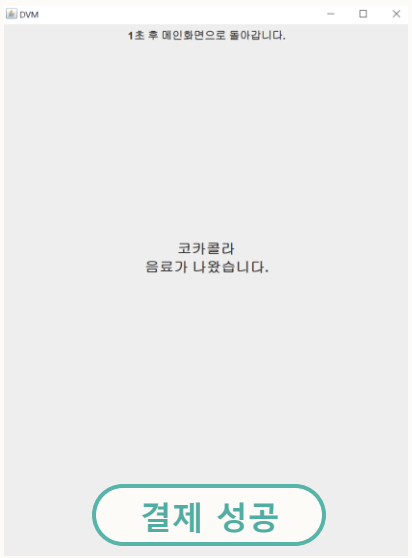


Touch!

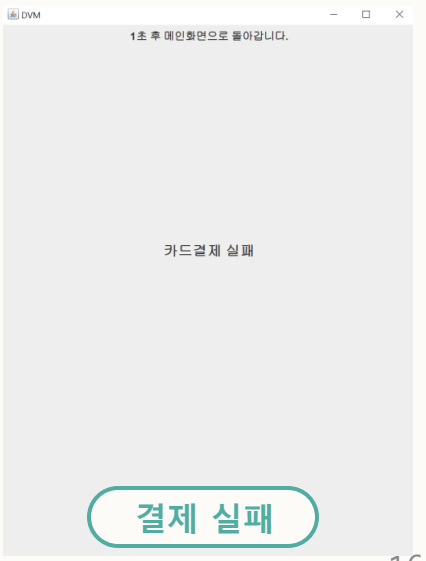


재고가 있는 제품 선택

카드결제



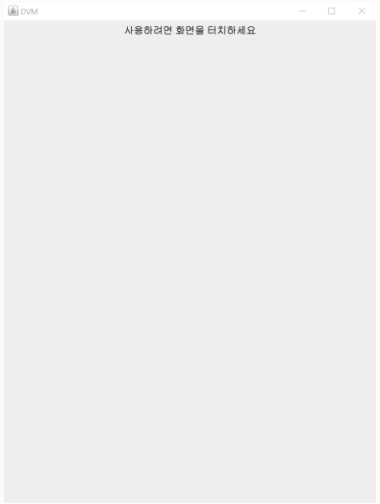
결제 성공



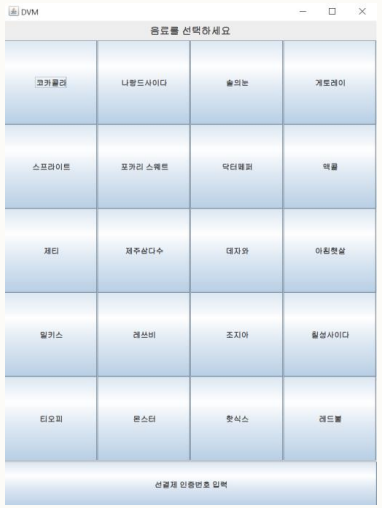
결제 실패

Activity 2052. Implement Windows (사용자 매뉴얼)

첫 실행화면



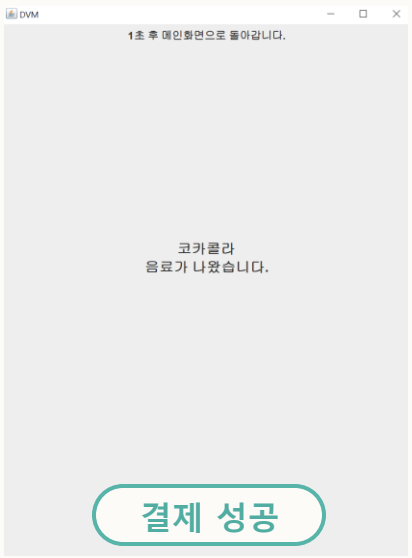
Touch!



재고가 있는 제품 선택

간편결제

Touch!

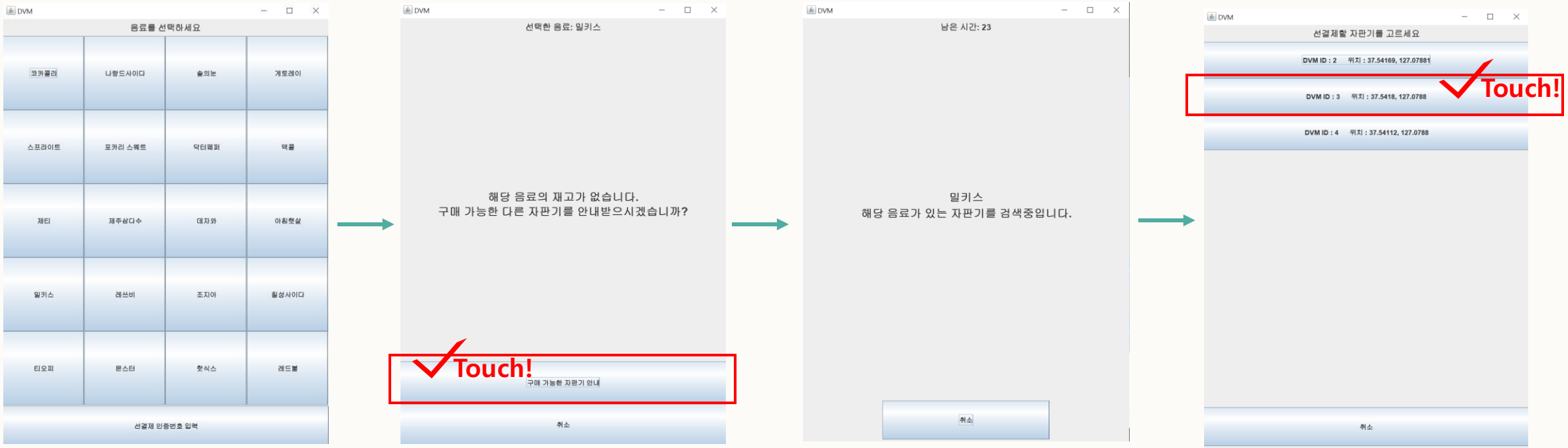


결제 성공

결제 실패

Activity 2052. Implement Windows (사용자 매뉴얼)

재고가 존재하지 않는 제품 선택



Activity 2052. Implement Windows (사용자 매뉴얼)

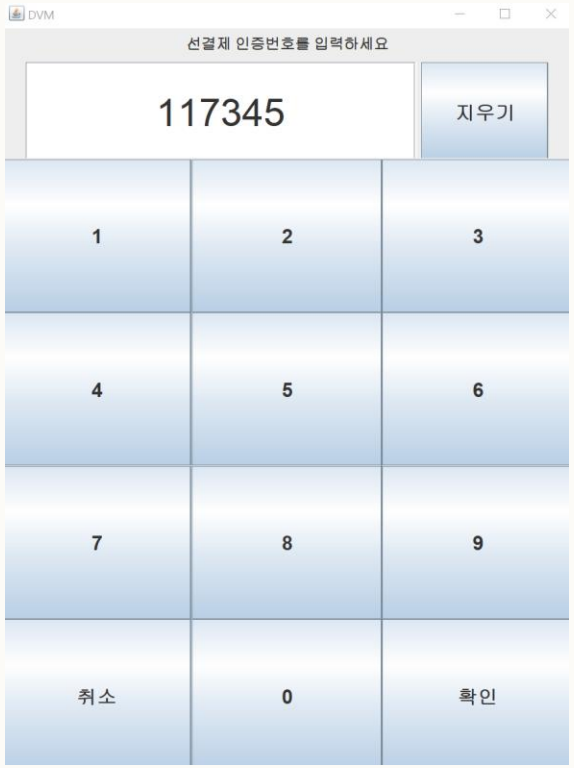
재고가 존재하지 않는 제품 선택

선결제 성공

선결제 실패

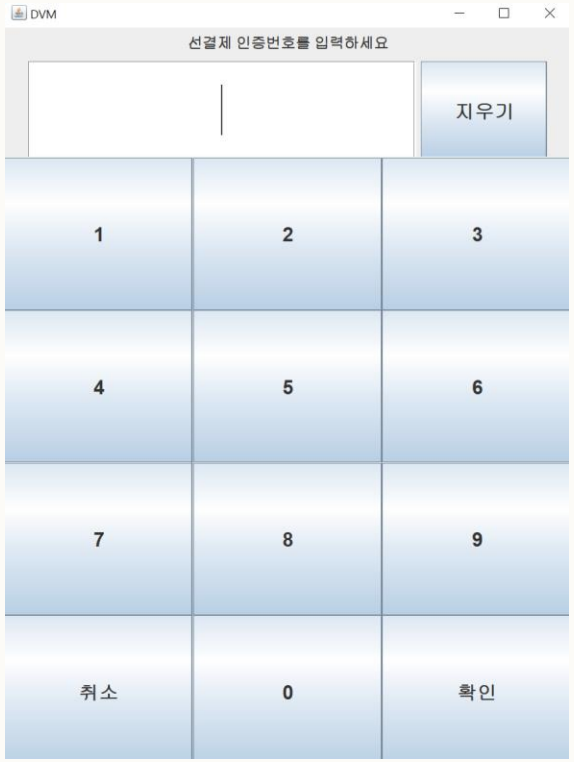
Activity 2052. Implement Windows (사용자 매뉴얼)

선결제 된 인증번호 입력



Activity 2052. Implement Windows (사용자 매뉴얼)

존재하지 않는 인증번호 입력



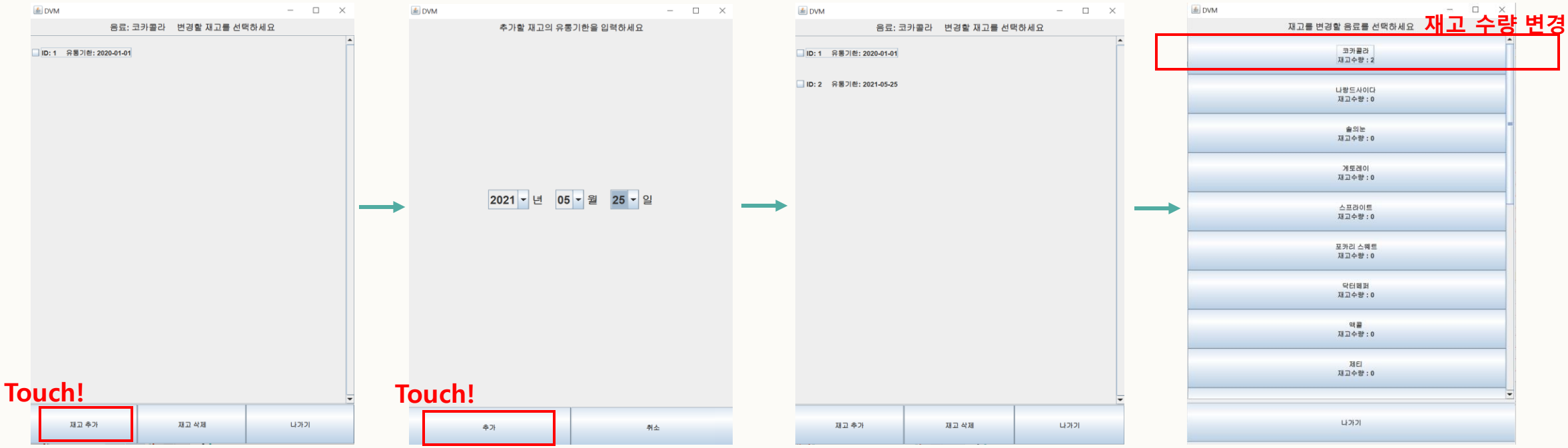
Activity 2052. Implement Windows (사용자 매뉴얼)

관리자 모드

관리자 고유번호 입력

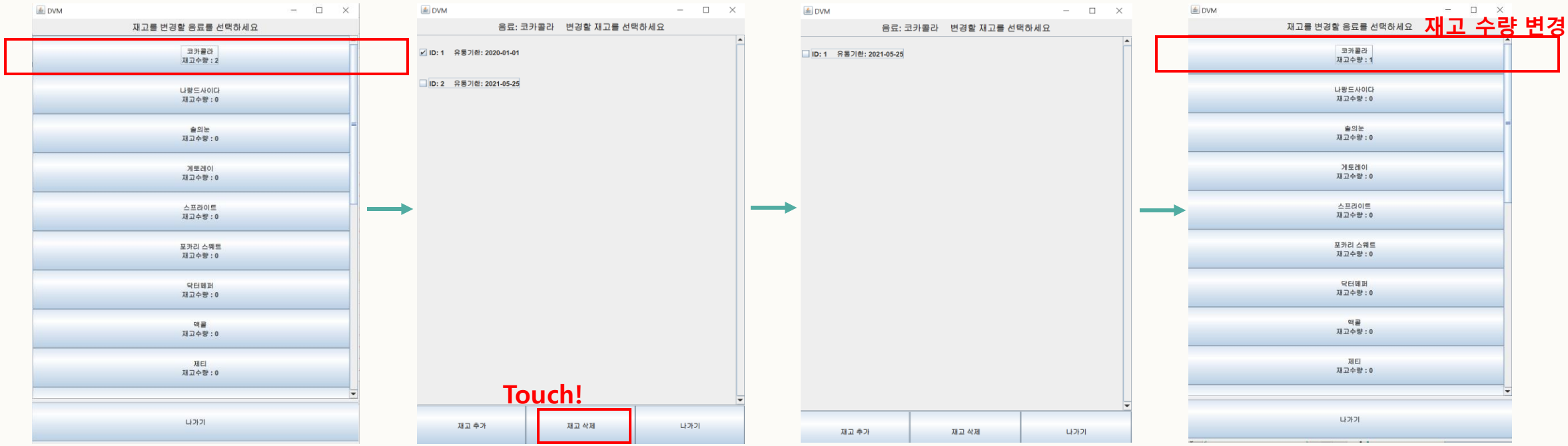
Activity 2052. Implement Windows (사용자 매뉴얼)

재고 추가



Activity 2052. Implement Windows (사용자 매뉴얼)

재고 삭제



Activity 2055. Write Unit Test Code

Class:Title

```java
package Logic;

import org.junit.Assert;
import org.junit.Test;

import java.util.ArrayList;

import static org.junit.Assert.*;

public class TitleTest {
    private final Title t = new Title( Name: "코카콜라",  Price: 700);

    @Test
    public void testAddItem() {
        t.AddItem(new Item( Expiration_Date: 20201125));
        t.AddItem(new Item( Expiration_Date: 20201125));
        t.AddItem(new Item( Expiration_Date: 20201125));
        t.AddItem(new Item( Expiration_Date: 20201125));
        t.AddItem(new Item( Expiration_Date: 20201125));
        int ExpectedResult = 5;
        int ActualResult = t.getItem_List().size();
        Assert.assertEquals(ExpectedResult, ActualResult);
    }
```

```java
    @Test
    public void testDeleteItem() {
        t.AddItem(new Item( Expiration_Date: 20201125));
        t.AddItem(new Item( Expiration_Date: 20201125));
        t.AddItem(new Item( Expiration_Date: 20201125));
        t.AddItem(new Item( Expiration_Date: 20201125));
        t.AddItem(new Item( Expiration_Date: 20201125));
        ArrayList<Integer> delitemlist = new ArrayList<>();
        delitemlist.add(0);
        delitemlist.add(1);
        delitemlist.add(2);
        t.DeleteItem(delitemlist);
        int ExpectedResult = 2;
        int ActualResult = t.getItem_List().size();
        Assert.assertEquals(ExpectedResult, ActualResult);
    }

    @Test
    public void testItem_List() {
        t.AddItem(new Item( Expiration_Date: 20201025));
        t.AddItem(new Item( Expiration_Date: 20201125));
        t.AddItem(new Item( Expiration_Date: 20201125));
        t.AddItem(new Item( Expiration_Date: 20201125));
        t.AddItem(new Item( Expiration_Date: 20201125));
        int ExpectedResult = 20201025;
        int ActualResult = t.getItem_List().get(0).getExpiration_Date();
        Assert.assertEquals(ExpectedResult, ActualResult);
    }
}
```

Activity 2055. Write Unit Test Code

Class:Title

```
54      @Test
55      public void testUpdateStock() {
56          t.AddItem(new Item( Expiration_Date: 20201025));
57          t.AddItem(new Item( Expiration_Date: 20201125));
58          t.AddItem(new Item( Expiration_Date: 20201125));
59          t.AddItem(new Item( Expiration_Date: 20201125));
60          t.AddItem(new Item( Expiration_Date: 20201125));
61          t.UpdateStock( id: 1,  IfHold: true);
62          int ExpectedResult = 1;
63          int ActualResult = t.getHold();
64          Assert.assertEquals(ExpectedResult, ActualResult);
65          ExpectedResult = 4;
66          ActualResult = t.getItem_List().size();
67          Assert.assertEquals(ExpectedResult, ActualResult);
68          t.UpdateStock( id: 1,  IfHold: false);
69          ExpectedResult = 1;
70          ActualResult = t.getHold();
71          Assert.assertEquals(ExpectedResult, ActualResult);
72          ExpectedResult = 4;
73          ActualResult = t.getItem_List().size();
74          Assert.assertEquals(ExpectedResult, ActualResult);
75          t.UpdateStock( id: 0,  IfHold: true);
76          ExpectedResult = 0;
77          ActualResult = t.getHold();
78          Assert.assertEquals(ExpectedResult, ActualResult);
79          ExpectedResult = 4;
80          ActualResult = t.getItem_List().size();
81          Assert.assertEquals(ExpectedResult, ActualResult);
82          t.UpdateStock( id: 0,  IfHold: false);
83          ExpectedResult = 0;
84          ActualResult = t.getHold();
85          Assert.assertEquals(ExpectedResult, ActualResult);
86          ExpectedResult = 3;
```

```
83          ExpectedResult = 0;
84          ActualResult = t.getHold();
85          Assert.assertEquals(ExpectedResult, ActualResult);
86          ExpectedResult = 3;
87          ActualResult = t.getItem_List().size();
88          Assert.assertEquals(ExpectedResult, ActualResult);
89      }
90
91      @Test
92      public void testCheckStock() {
93          t.AddItem(new Item( Expiration_Date: 20201025));
94          assertTrue(t.CheckStock());
95          t.UpdateStock( id: 0,  IfHold: false);
96          assertFalse(t.CheckStock());
97      }
98  }
```

Activity 2055. Write Unit Test Code

Class:Item

```java
package Logic;

import org.junit.Assert;
import org.junit.Test;

import static org.junit.Assert.*;

public class ItemTest {

    Item item = new Item( Expiration_Date: 20201125);
    ;

    @Test
    public void testGetExpiration_Date() {
        item.setExpiration_Date(20211026);
        int ExpectedResult = 20211026;
        int ActualResult = item.getExpiration_Date();
        Assert.assertEquals(ExpectedResult, ActualResult);
    }
}
```

Activity 2055. Write Unit Test Code

Class:Payment

```java
package Logic;

import org.junit.Assert;
import org.junit.Test;

import static org.junit.Assert.*;

public class PaymentTest {

    Payment payment = new Payment( title_id: 1,  DVMid: 1);
    ;

    @Test
    public void testCardPay() {
        int ActualResult = payment.CardPay( Pay_confirm: true);
        Assert.assertTrue( condition: ActualResult > 0);
        int ExpectedResult = -3;
        ActualResult = payment.CardPay( Pay_confirm: false);
        Assert.assertEquals(ExpectedResult, ActualResult);
        payment = new Payment( title_id: 1,  DVMid: 0);
        ExpectedResult = 0;
        ActualResult = payment.CardPay( Pay_confirm: true);
        Assert.assertEquals(ExpectedResult, ActualResult);
        ExpectedResult = -3;
        ActualResult = payment.CardPay( Pay_confirm: false);
        Assert.assertEquals(ExpectedResult, ActualResult);
    }
```

```java
    @Test
    public void testSmartPay() {
        int ActualResult = payment.SmartPay( Pay_confirm: true);
        Assert.assertTrue( condition: ActualResult > 0);
        int ExpectedResult = -3;
        ActualResult = payment.SmartPay( Pay_confirm: false);
        Assert.assertEquals(ExpectedResult, ActualResult);
        payment = new Payment( title_id: 1,  DVMid: 0);
        ExpectedResult = 0;
        ActualResult = payment.SmartPay( Pay_confirm: true);
        Assert.assertEquals(ExpectedResult, ActualResult);
        ExpectedResult = -3;
        ActualResult = payment.SmartPay( Pay_confirm: false);
        Assert.assertEquals(ExpectedResult, ActualResult);
    }

    @Test
    public void testInit() {
        payment = new Payment( title_id: 1,  DVMid: 1);
        payment.init();
        int ExpectedResult = -1;
        int ActualResult = payment.getTitle_id();
        Assert.assertEquals(ActualResult, ExpectedResult);
        ExpectedResult = -1;
        ActualResult = payment.getDVMid();
        Assert.assertEquals(ExpectedResult, ActualResult);
        assertNull(payment.getError_log());
    }
}
```

Activity 2055. Write Unit Test Code

Class:C_Number

```java
package Logic;

import org.junit.Assert;
import org.junit.Test;

import static org.junit.Assert.*;

public class C_NumberTest {

    private C_Number CN = new C_Number( title_id: 1,  id: 1);

    @Test
    public void testCreateCnumber() {
        int ExpectedResult = CN.CreateCnumber( title_id: 1,  DvmID: 1);
        int ActualResult = CN.getC_Number_t();
        Assert.assertEquals(ExpectedResult, ActualResult);
    }
}
```

Activity 2055. Write Unit Test Code

Class: C_NumberManager

```java
package Logic;

import org.junit.Assert;
import org.junit.Test;

import static org.junit.Assert.*;

public class C_NumberManagerTest {
    private C_NumberManager CM= new C_NumberManager();
    private C_Number Cn=new C_Number( title_id: 1, id: 1);

    @Test
    public void testPopCnumber() {
        Cn.setC_Number_t(971125);
        CM.AddCnumber(Cn);
        int ExpectedResult=Cn.getTitle_id();
        int ActualResult=CM.PopCnumber( C_Number_t: 971125);
        Assert.assertEquals(ExpectedResult,ActualResult);
    }
```

```java
    @Test
    public void testCheckCnumber() {
        Cn.setC_Number_t(999999);
        CM.setM_Number(971125);
        CM.AddCnumber(Cn);
        int ExpectedResult=1;
        int ActualResult=CM.CheckCnumber( C_Number_t: 999999);
        Assert.assertEquals(ExpectedResult,ActualResult);
        ExpectedResult=0;
        ActualResult=CM.CheckCnumber( C_Number_t: 971125);
        Assert.assertEquals(ExpectedResult,ActualResult);
        ExpectedResult=-1;
        ActualResult=CM.CheckCnumber( C_Number_t: 888888);
        Assert.assertEquals(ExpectedResult,ActualResult);
    }

    @Test
    public void testAddCnumber() {
        Cn.setC_Number_t(971125);
        CM.AddCnumber(Cn);
        int ExpectedResult = 1;
        int ActualResult = CM.getC_List().size();
        Assert.assertEquals(ExpectedResult,ActualResult);
    }
}
```

Activity 2055. Write Unit Test Code

Class: DVM

```java
package Logic;

import org.junit.Assert;
import org.junit.Test;

import static org.junit.Assert.*;

public class DVMTest {
    private DVM dvm = new DVM( ID: 1,  X: 1.0,  Y: 1.0);

    @Test
    public void testGetID() {
        dvm.setID(7);
        int ExpectedResult = 7;
        int ActualResult = dvm.getID();
        Assert.assertEquals(ExpectedResult, ActualResult);
    }

    @Test
    public void testGetAddress_X() {
        dvm.setAddress_X(1.23456);
        Double ExpectedResult = 1.23456;
        Double ActualResult = dvm.getAddress_X();
        Assert.assertEquals(ExpectedResult, ActualResult);
    }

    @Test
    public void testGetAddress_Y() {
        dvm.setAddress_Y(2.45678);
        Double ExpectedResult = 2.45678;
        Double ActualResult = dvm.getAddress_Y();
        Assert.assertEquals(ExpectedResult, ActualResult);
    }
```

```java
    public void testGetCurrentID() {
        dvm.CurrentID=2;
        Assert.assertEquals( expected: 2,dvm.CurrentID);
    }
}
```

Activity 2055. Write Unit Test Code

Class:Message

```java
package Logic;

import org.junit.Assert;
import org.junit.Test;

import static org.junit.Assert.*;

public class MessageTest {
    Message msg = new Message( myid: 1);

    @Test
    public void testSetmsg1() {
        msg.setmsg( id: 1,  type: 1,  data: 1);
        Assert.assertEquals( expected: 1, msg.getTargetID());
        Assert.assertEquals( expected: 1, msg.getType());
        Assert.assertEquals( expected: 1, msg.getTitle());
    }

    @Test
    public void testSetmsg2() {
        msg.setmsg( id: 1,  type: 2,  data: true);
        Assert.assertEquals( expected: 1, msg.getTargetID());
        Assert.assertEquals( expected: 2, msg.getType());
        Assert.assertTrue(msg.isBoolData());
    }

    @Test
    public void testSetmsg3() {
        msg.setmsg( id: 1,  type: 3);
        Assert.assertEquals( expected: 1, msg.getTargetID());
        Assert.assertEquals( expected: 3, msg.getType());
    }
```

```java
    @Test
    public void testSetmsg4() {
        msg.setmsg( id: 1,  type: 4,  data1: 1.0,  data2: 1.0);
        Assert.assertEquals( expected: 1, msg.getTargetID());
        Assert.assertEquals( expected: 4, msg.getType());
        Assert.assertEquals(Double.toString( d: 1.0), Double.toString(msg.getxAdress()));
        Assert.assertEquals(Double.toString( d: 1.0), Double.toString(msg.getyAdress()));
    }

    @Test
    public void testSetmsg5() {
        msg.setmsg( id: 1,  type: 5,  data1: 1,  data2: 971125);
        Assert.assertEquals( expected: 1, msg.getTargetID());
        Assert.assertEquals( expected: 5, msg.getType());
        Assert.assertEquals( expected: 1, msg.getTitle());
        Assert.assertEquals( expected: 971125, msg.getC_Number());


    }
}
```

Activity 2055. Write Unit Test Code

Class:Message_Queue

```java
package Logic;

import org.junit.Assert;
import org.junit.Test;

import java.net.InetAddress;

public class Message_QueueTest {
    private Message msg = new Message( myid: 1);
    private Message_Queue queue = new Message_Queue();

    @Test
    public void testMsgSend() {
        msg.setTargetID(1);
        msg.setType(2);
        msg.setBoolData(true);
        Message_Queue.MsgSend(msg);
        msg.setType(3);
        Message_Queue.MsgSend(msg);
    }

    @Test
    public void testDequeue() {
        Controller c = new Controller();
        c.getTitle_List().get(0).AddItem(new Item( Expiration_Date: 20201125));
        msg.setTargetID(1);
        msg.setType(5);
        msg.setTitle(1);
        msg.setC_Number(971125);
        Message_Queue.msgQueue.offer(msg);
        Message_Queue.Dequeue();
    }
}
```

Activity 2055. Write Unit Test Code

Class:Controller

```java
package Logic;

import GUI.*;
import org.junit.Assert;
import org.junit.Before;
import org.junit.Ignore;
import org.junit.Test;

import javax.swing.*;

import static org.junit.Assert.*;

public class ControllerTest {
    private Controller c = new Controller();

    @Ignore
    @Before
    public void initController() {
        c = new Controller();
        c.setK(new JFrame());
    }

    @Ignore
    @Test
    public void inputSelect() {
        int del;
        c.setK(new MainMenu(c.getTitle_List()));
        del = c.InputSelect();
        Assert.assertEquals( expected: -2, del);
        c.getK().setVisible(false);
    }

    @Ignore
    @Test
    public void showInputLine() {
        int del;
        c.setK(new MainMenu(c.Title_List));
        del = c.InputSelect();
        Assert.assertEquals( expected: 0, del);
        c.getK().setVisible(false);
    }

    @Ignore
    @Test
    public void inputCnumber() {
        int del;
        c.setK(new InputLine());
        del = c.InputCnumber();
        Assert.assertEquals( expected: 0, del);
        c.getK().setVisible(false);
    }

    @Ignore
    @Test
    public void infoCnumberError() {
        int del;
        c.setK(new InputLine());
        del = c.InputCnumber();
        Assert.assertEquals( expected: 1, del);
        c.getK().setVisible(false);
    }
```

Activity 2055. Write Unit Test Code

Class:Controller

```
65  public void manShowTitle() {
66      c.ManShowTitle();
67      Assert.assertEquals( expected: 0, c.getDel());
68      c.getK().setVisible(false);
69  }
70
71  @Ignore
72  @Test
73  public void manSelectTitle() {
74      c.ManShowTitle();
75      Assert.assertEquals( expected: 0, c.getDel());
76      c.getK().setVisible(false);
77  }
78
79  @Ignore
80  @Test
81  public void manShowItem() {
82      c.Title_List.get(0).AddItem(new Item( Expiration_Date: 20250101));
83      c.ManShowItem( TitleID: 1);
84      Assert.assertEquals( expected: 0, c.getDel());
85      c.getK().setVisible(false);
86  }
87
88  @Ignore
89  @Test
90  public void manEditItem() {
91      c.Title_List.get(0).AddItem(new Item( Expiration_Date: 20250101));
92      c.ManShowItem( TitleID: 1);
93      Assert.assertEquals( expected: 2, c.getTitle_List().get(0).getItem_List().size());
94      c.ManShowItem( TitleID: 1);
95      Assert.assertEquals( expected: 0, c.getTitle_List().get(0).getItem_List().size());
96      c.getK().setVisible(false);
97  }
```

```
99   @Ignore
100  @Test
101  public void printSelectPay() {
102      c.setK(new MainMenu(c.Title_List));
103      c.Title_List.get(0).AddItem(new Item( Expiration_Date: 20201125));
104      c.InputSelect();
105      Assert.assertEquals( expected: 1, c.Title_List.get(0).getItem_List().size());
106      c.getK().setVisible(false);
107  }
108
109  @Ignore
110  @Test
111  public void selectPayment() {
112      c.setK(new PaymentMenu(new Title( Name: "test",  Price: 700)));
113      c.setBasket(1);
114      c.SelectPayment( DVMid: 0);
115      Assert.assertEquals( expected: 0, c.getDel());
116      c.getK().setVisible(false);
117  }
118
119  @Ignore
120  @Test
121  public void cancelItem() {
122      c.setBasket(1);
123      c.setPayment(new Payment( title_id: 1,  DVMid: 0));
124      c.CancelItem();
125      Assert.assertEquals( expected: -666, c.getBasket());
126      Assert.assertTrue(c.DVMStack.isEmpty());
127      Assert.assertNull(c.getPayment());
128      c.getK().setVisible(false);
129  }
```

Activity 2055. Write Unit Test Code

Class:Controller

```
131     @Ignore
132     @Test
133     public void showCardPay() {
134         c.setPayment(new Payment( title_id: 1,  DVMid: 0));
135         c.ShowCardPay();
136         c.getK().setVisible(false);
137     }
138
139     @Ignore
140     @Test
141     public void cardPay() {
142         c.setK(new CardPayUI());
143         c.Title_List.get(0).AddItem(new Item( Expiration_Date: 20201125));
144         c.setBasket(1);
145         c.setPayment(new Payment( title_id: 1,  DVMid: 0));
146         c.CardPay();
147         Assert.assertEquals( expected: 0, c.getTitle_List().get(0).getItem_List().size());
148         c.getK().setVisible(false);
149     }
150
151     @Ignore
152     @Test
153     public void showSmartPay() {
154         c.setPayment(new Payment( title_id: 1,  DVMid: 0));
155         c.ShowSmartPay();
156         c.getK().setVisible(false);
157     }
158
```

```
159     @Ignore
160     @Test
161     public void smartPay() {
162         c.setK(new SmartPayUI());
163         c.Title_List.get(0).AddItem(new Item( Expiration_Date: 20201125));
164         c.setBasket(1);
165         c.setPayment(new Payment( title_id: 1,  DVMid: 0));
166         c.SmartPay();
167         Assert.assertEquals( expected: 0, c.Title_List.get(0).getItem_List().size());
168         c.getK().setVisible(false);
169     }
170
171     @Ignore
172     @Test
173     public void returnItem() {
174         c.Title_List.get(0).AddItem(new Item( Expiration_Date: 20201125));
175         c.setBasket(1);
176         c.setPayment(new Payment( title_id: 1,  DVMid: 0));
177         c.ReturnItem(c.Title_List.get(0),  ifHold: false);
178         Assert.assertEquals( expected: 0, c.Title_List.get(0).getItem_List().size());
179         c.getK().setVisible(false);
180     }
181
182     @Ignore
183     @Test
184     public void printCnumber() {
185         c.setBasket(1);
186         c.setPayment(new Payment( title_id: 1,  DVMid: 1));
187         c.PrintCnumber(c.Title_List.get(0),  DVMid: 1,  Cnumber: 66666);
188         c.getK().setVisible(false);
189     }
```

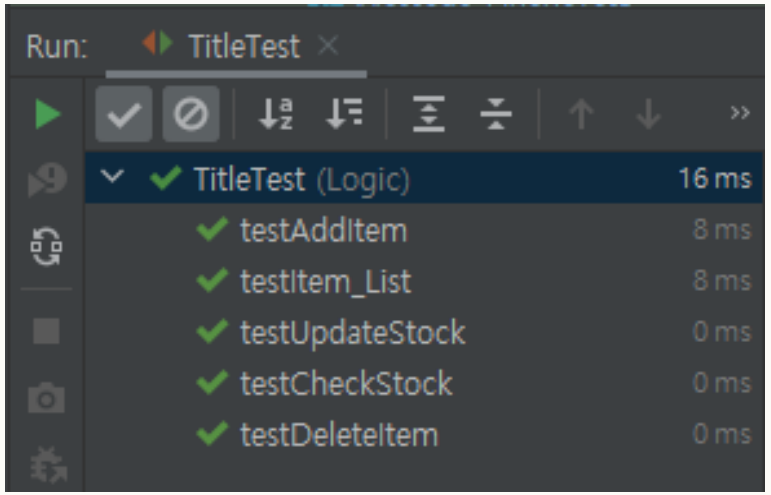
Activity 2055. Write Unit Test Code

Class:Controller

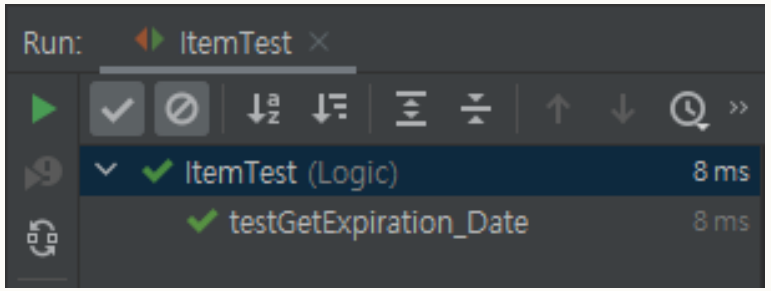
```
191     @Ignore
192     @Test
193     public void infoNoItem() {
194         c.setBasket(1);
195         c.InfoNoItem();
196         c.getK().setVisible(false);
197     }
198
199     @Ignore
200     @Test
201     public void reqFindDVM() {
202         c.setBasket(1);
203         c.InfoNoItem();
204         c.getK().setVisible(false);
205     }
206
207     @Test
208     public void testInit() {
209         c.setBasket(10);
210         c.setPayment(new Payment(c.getBasket(),  DVMid: 1));
211         c.init();
212         int ExpectedResult = -666;
213         int ActualResult = c.getBasket();
214         Assert.assertEquals(ExpectedResult, ActualResult);
215         Assert.assertNull(c.getPayment());
216     }
217 }
```

Activity 2061. Unit Testing

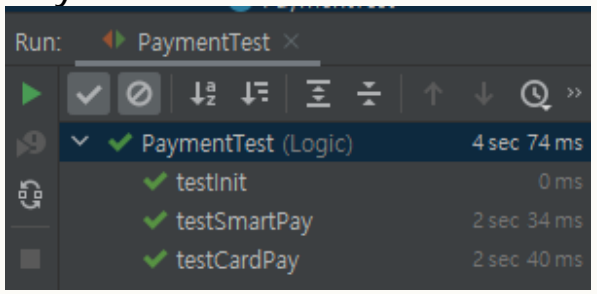
TitleTest



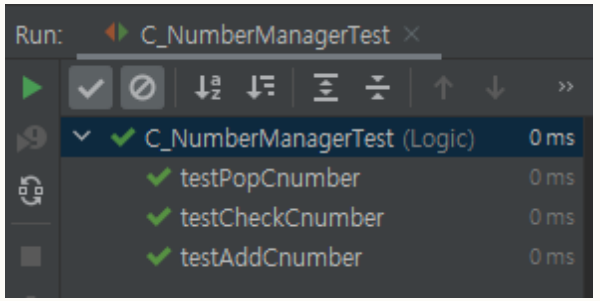
ItemTest



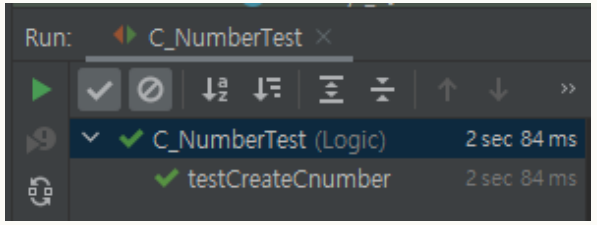
PaymentTest



C_NumberManagerTest

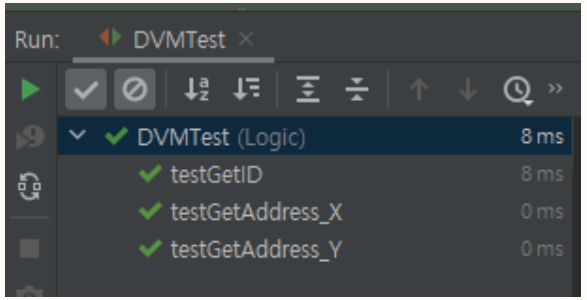


C_NumberTest

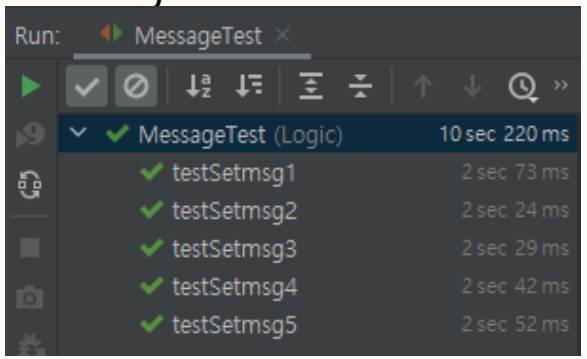


Activity 2061. Unit Testing

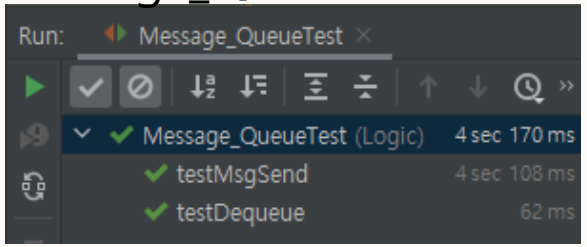
DVMTest



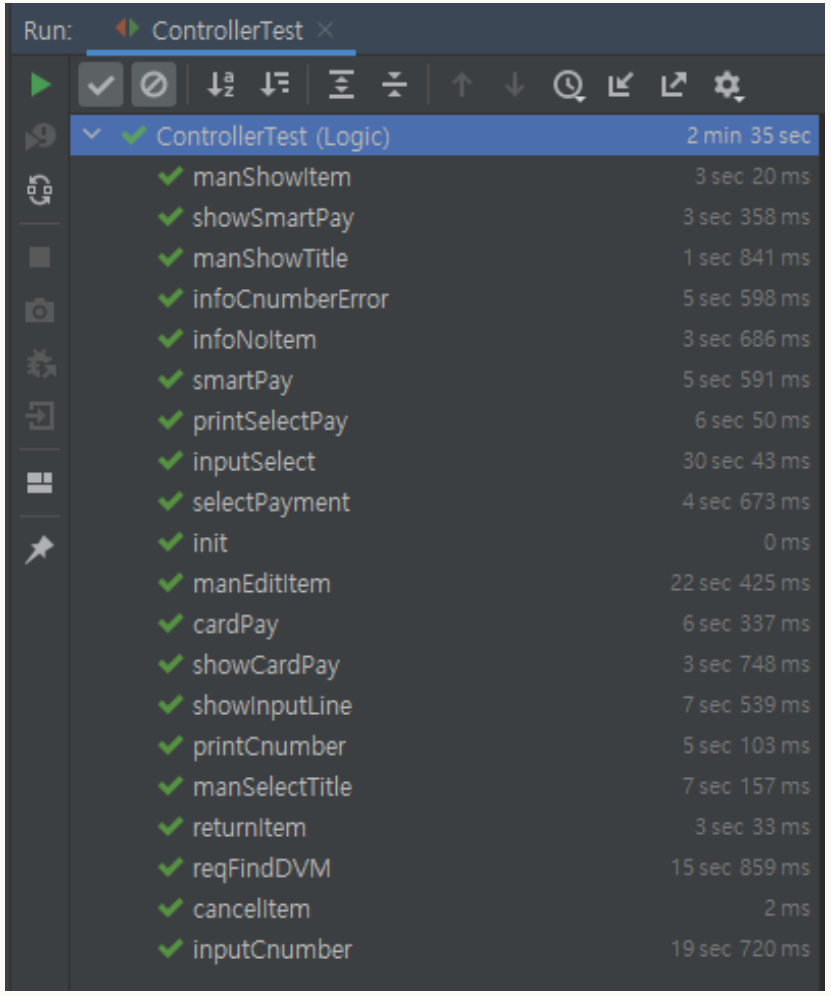
MessageTest



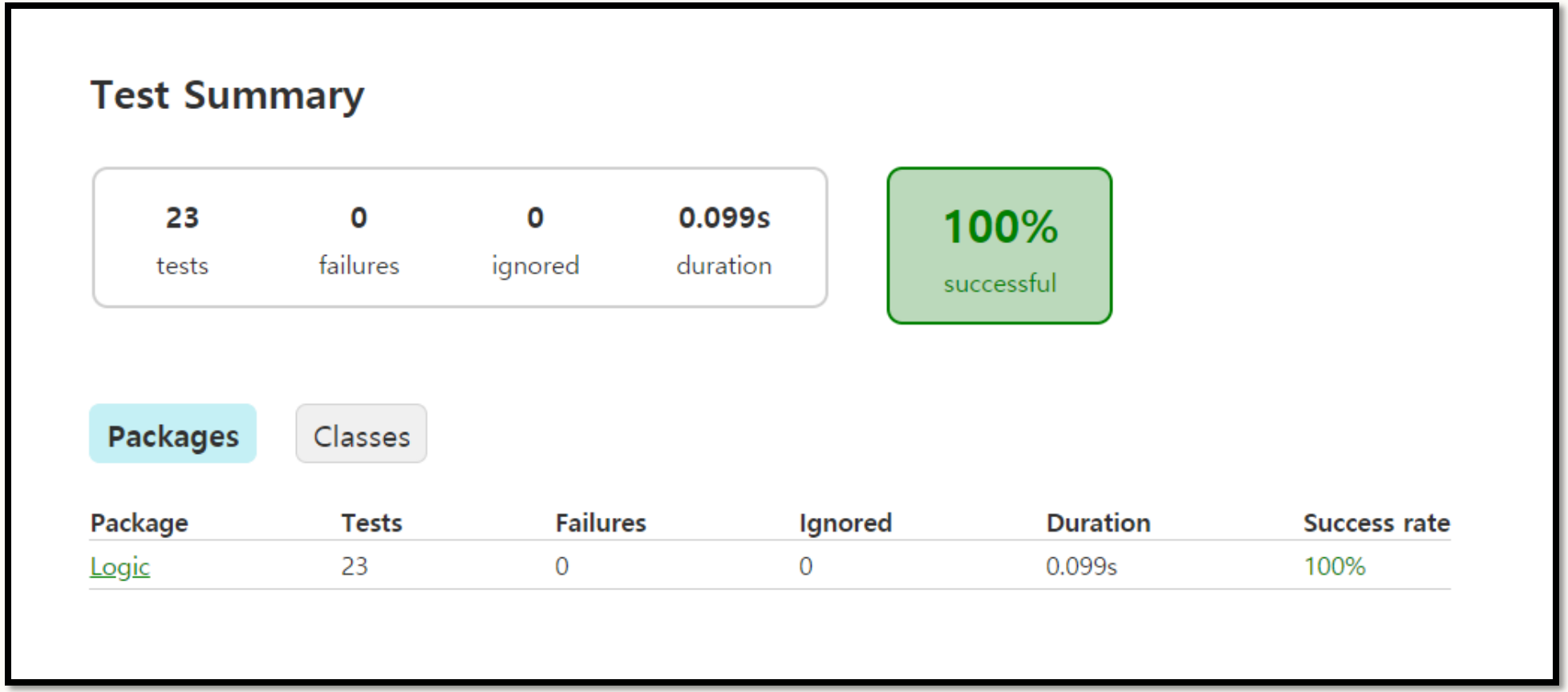
Message_QueueTest



ControllerTest



Activity 2061. Unit Testing



Test Summary

23	0	0	0.099s
tests	failures	ignored	duration

100%
successful

Packages | Classes

Package	Tests	Failures	Ignored	Duration	Success rate
Logic	23	0	0	0.099s	100%

Activity 2063. System Testing

Ref	Use Case name	Test Description	P/F
R1.1	1. 자판기 음료 종류 출력	- 20가지 메뉴가 정상적으로 출력되는지 확인	**P**
R1.2.1	2. 구매할 음료 입력	- 사용자가 선택한 음료가 정확히 기록되는지 확인	**P**
R.1.2.2	3. 사용자 선택 취소	- 사용자 선택 취소 시 메인 화면으로 돌아가는 지 확인	**P**
R1.2.3	4. 사용자 인증번호/바코드 입력	- 다른 인증번호가 입력된 경우 오류 메시지를 출력하는지 확인	**P**
		- 올바른 인증번호가 입력된 경우 보관된 제품 중 인증번호에 맞는 제품을 전달하는지 확인	**P**
R1.3	5. 재고가 부족한 제품 안내	- 재고가 부족한 제품을 정확히 출력하는지 확인	**P**
R1.4	6. 구매 가능한 자판기 안내	- 정확한 위치정보를 출력하는지 확인	**P**
		- 지정된 물품의 재고가 정확히 존재하는지 확인	**P**
		- 지정된 물품의 재고가 존재하는 자판기가 모두 출력되는지 확인	**P**

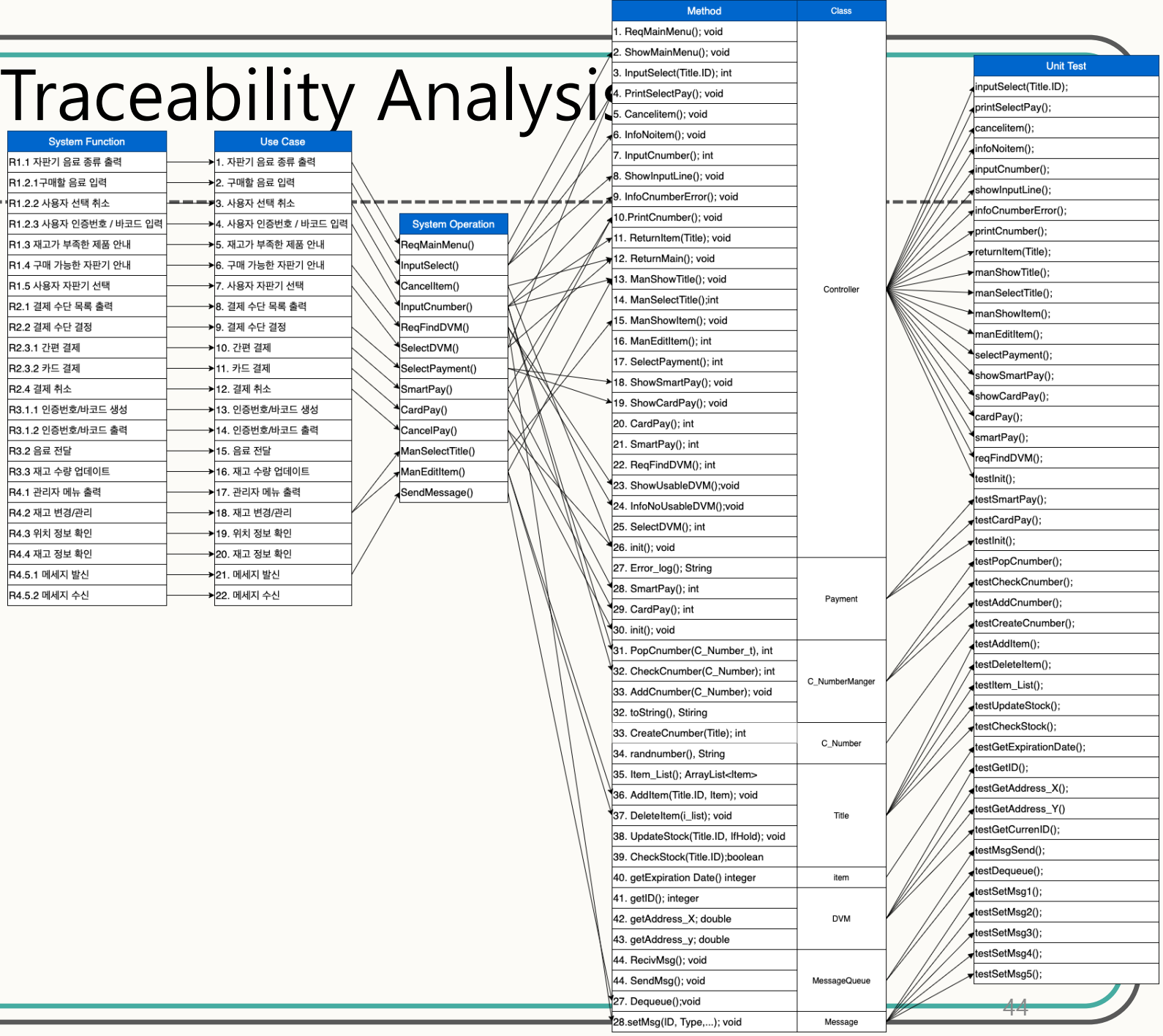
Activity 2063. System Testing

Ref	Use Case name	Test Description	P/F
R1.5	7. 사용자 자판기 선택	- 사용자가 선택한 DVM이 정확히 기록되는 지 확인	**P**
R2.1	8. 결제 수단 목록 출력	- 사용자가 선택한 음료가 정확히 목록화되어 있는지 확인	**P**
R2.2	9. 결제 수단 결정	- 사용자가 선택한 결제 수단에 따른 결제 화면으로 넘어가는지 확인	**P**
R2.3.1	10. 간편 결제	- 사용자가 선택한 제품들이 정상적으로 합산되어 계산 되는지 확인	**P**
R2.3.2	11. 카드 결제		
R2.4	12. 결제 취소	- 결제 취소 시 메인 화면으로 돌아가는 지 확인	**P**
R3.1.1	13. 인증번호/바코드 생성	- 인증번호/바코드가 중복되지 않게 생성되는지 확인	**P**
R3.1.2	14. 인증번호/바코드 출력	- 생성된 인증번호/바코드가 정상적으로 출력되는지 확인	**P**

Activity 2063. System Testing

Ref	Use Case name	Test Description	P/F
R3.2	15. 음료 전달	- 사용자가 선택한 제품들이 합산되어 전달 되는지 확인	**P**
R3.3	16. 재고 수량 업데이트	- 사용자 이용 이후 변화된 재고가 정확히 변화하는지 확인	**P**
R4.1	17. 관리자 메뉴 출력	- 정확한 절차가 시행됨에 따라 메뉴가 출력 되는지 확인	**P**
R4.2	18. 재고 변경/관리	- 관리자의 동작에 따라 재고의 수량이 잘 변경되는지 확인	**P**
R4.3	19. 위치 정보 확인	- 정확한 위치 정보를 반환하는지 확인	**P**
R4.4	20. 재고 정보 확인	- 정확한 재고 정보를 반환하는지 확인	**P**
R4.5.1	21. 메시지 발신	- DVM네트워크를 통한 메시지가 정확히 목표 DVM으로 전송되는지 확인	**P**
R4.5.2	22. 메시지 수신	- DVM네트워크를 통한 메시지가 정확히 목표 DVM으로 수신되는지 확인	**P**

Activity 2067. Testing Traceability Analysis



Demonstration Video

https://youtu.be/1VZMeKxbdWw